Blind Justice: Fairness with Encrypted Sensitive Attributes

Niki Kilbertus [1 2] **Adrià Gascón** [3 4] **Matt Kusner** [3 4] **Michael Veale** [5] **Krishna P. Gummadi** [6] **Adrian Weller** [2 3]

Abstract

Recent work has explored how to train machine learning models which do not discriminate against any subgroup of the population as determined by sensitive attributes such as gender or race. To avoid disparate treatment, sensitive attributes should not be considered. On the other hand, in order to avoid disparate impact, sensitive attributes must be examined—e.g., in order to learn a fair model, or to check if a given model is fair. We introduce methods from secure multi-party computation which allow us to avoid both. By encrypting sensitive attributes, we show how an outcome-based fair model may be learned, checked, or have its outputs verified and held to account, *without users revealing their sensitive attributes*.

1. Introduction

Concerns are rising that machine learning systems which make or support important decisions affecting individuals—such as car insurance pricing, résumé filtering or recidivism prediction—might illegally or unfairly discriminate against certain subgroups of the population (Schreurs et al., 2008; Calders & Žliobaitė, 2012; Barocas & Selbst, 2016). The growing field of *fair learning* seeks to formalize relevant requirements, and through altering parts of the algorithmic decision-making pipeline, to detect and mitigate potential discrimination (Friedler et al., 2016).

Most legally-problematic discrimination centers on differences based on *sensitive attributes*, such as gender or race (Barocas & Selbst, 2016). The first type, *disparate treatment* (or *direct discrimination*), occurs if individuals are treated differently according to their sensitive attributes (with all others equal). To avoid disparate treatment, one should not inquire about individuals' sensitive attributes. While this has some intuitive appeal and justification (Grgić-Hlača et al., 2018), a significant concern is that sensitive attributes may often be accurately predicted ("reconstructed") from non-sensitive features (Dwork et al., 2012). This motivates measures to deal with the second type of discrimination.

Disparate impact (or *indirect discrimination*) occurs when the *outcomes* of decisions disproportionately benefit or hurt individuals from subgroups with particular sensitive attribute settings without appropriate justification. For example, firms deploying car insurance telematics devices (Handel et al., 2014) build up high dimensional pictures of driving behavior which might easily proxy for sensitive attributes even when they are omitted. Much recent work in fair learning has focused on approaches to avoiding various notions of disparate impact (Feldman et al., 2015; Hardt et al., 2016; Zafar et al., 2017c).

In order to check and enforce such requirements, the modeler must have access to the sensitive attributes for individuals in the training data—however, this may be undesirable for several reasons (Žliobaitė & Custers, 2016). First, individuals are unlikely to want to entrust sensitive attributes to modelers in all application domains. Where applications have clear discriminatory potential, it is understandable that individuals may be wary of providing sensitive attributes to modelers who might exploit them to negative effect, especially with no guarantee that a fair model will indeed be learned and deployed. Even if certain modelers themselves were trusted, the wide provision of sensitive data creates heightened privacy risks in the event of a data breach.

Further, legal barriers may limit collection and processing of sensitive personal data. A timely example is the EU's General Data Protection Regulation (GDPR), which contains heightened prerequisites for the collection and processing of some sensitive attributes. Unlike other data, modelers cannot justify using sensitive characteristics in fair learning with their "legitimate interests"—and instead will often need explicit, freely given consent (Veale & Edwards, 2018).

One way to address these concerns was recently proposed by Veale & Binns (2017). The idea is to involve a highly trusted third party, and may work well in some cases. However, there are significant potential difficulties: individuals must disclose their sensitive attributes to the third party (even if an individual trusts the party, she may have concerns that

[1]Max Planck Institute for Intelligent Systems [2]University of Cambridge [3]The Alan Turing Institute [4]University of Warwick [5]University College London [6]Max Planck Institute for Software Systems. Correspondence to: Niki Kilbertus <niki.kilbertus@tuebingen.mpg.de>.

Proceedings of the 35^{th} *International Conference on Machine Learning*, Stockholm, Sweden, PMLR 80, 2018.

the data may somehow be obtained or hacked by others, e.g., Graham, 2017); and the modeler must disclose their model to the third party, which may be incompatible with their intellectual property or other business concerns.

Contribution. We propose an approach to detect and mitigate disparate impact without disclosing readable access to sensitive attributes. This reflects the notion that decisions should be blind to an individual's status—depicted in courtrooms by a blindfolded Lady Justice holding balanced scales (Bennett Capers, 2012). We assume the existence of a regulator with fairness aims (such as a data protection authority or anti-discrimination agency). With recent methods from *secure multi-party computation* (MPC), we enable auditable fair learning while ensuring that both individuals' sensitive attributes and the modeler's model remain private to all other parties—including the regulator. Desirable fairness and accountability applications we enable include:

1. **Fairness certification.** Given a model and a dataset of individuals, check that the model satisfies a given fairness constraint (we consider several notions from the literature, see Section 2.2); if yes, generate a certificate.
2. **Fair model training.** Given a dataset of individuals, learn a model guaranteed and certified to be fair.
3. **Decision verification.** A malicious modeler might go through fair model training, but then use a different model in practice. To address such accountability concerns (Kroll et al., 2016), we efficiently provide for an individual to challenge a received outcome, verifying that it matches the outcome from the previously certified model.

We rely on recent theoretical developments in MPC (see Section 3) which we extend to admit linear constraints in order to enforce fairness requirements. These extensions may be of independent interest. We demonstrate the real-world efficacy of our methods, and shall make our code publicly available.

2. Fairness and Privacy Requirements

Here we formalize our setup and requirements.

2.1. Assumptions and Incentives

We assume three categories of participants: a *modeler* M, a *regulator* REG, and *users* $\mathrm{U}_1, \dots, \mathrm{U}_n$. For each user, we consider a vector of sensitive features (or attributes, we use the terms interchangeably) $\mathbf{z}_i \in \mathcal{Z}$ (e.g., ethnicity or gender) which might be a source of discrimination, and a vector of non-sensitive features $\mathbf{x}_i \in \mathcal{X}$ (discrete or real). Additionally, each user has a non-sensitive feature $y_i \in \mathcal{Y}$ which the modeler M would like to predict—the *label* (e.g., loan default). In line with current work in fair learning, we assume that all $\mathbf{z}_i$ and y_i attributes are binary, though our MPC approach could be extended to multi-label settings. The source of societal concern is that sensitive attributes $\mathbf{z}_i$ are potentially correlated with $\mathbf{x}_i$ or y_i.

Modeler M wishes to train a model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$, which accurately maps features $\mathbf{x}_i$ to labels y_i, in a supervised fashion. We assume M needs to keep the model private for intellectual property or other business reasons. The model $f_{\boldsymbol{\theta}}$ does not use sensitive information $\mathbf{z}_i$ as input to prevent disparate treatment (direct discrimination).

For each user U_i, M observes or is provided $\mathbf{x}_i, y_i$. The sensitive information in $\mathbf{z}_i$ is required to ensure $f_{\boldsymbol{\theta}}$ meets a given disparate impact fairness condition $\mathbb{F}$ (see Section 2.2). While each user U_i wants $f_{\boldsymbol{\theta}}$ to meet $\mathbb{F}$, they also wish to keep $\mathbf{z}_i$ private from all other parties. The regulator REG aims to ensure that M deploys only models that meet fairness condition $\mathbb{F}$. It has no incentive to collude with M (if collusion were a concern, more sophisticated cryptographic protocols would be required). Further, the modeler M might be legally obliged to demonstrate to the regulator REG that their model meets fairness condition $\mathbb{F}$ before it can be publicly deployed. As part of this, REG also has a positive duty to enable the training of fair models.

In Section 2.3, we define and address three fundamental problems in our setup: certification, training, and verification. For each problem, we present its functional goal and its privacy requirements. We refer to $\mathbf{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^n$ as the non-sensitive and sensitive data, respectively. In Section 2.2, we first provide necessary background on various notions of fairness that have been explored in the fair learning literature.

2.2. Fairness Criteria

In large part, works that formalize fairness in machine learning do so by balancing a certain condition between groups of people with different sensitive attributes, $\mathbf{z}$ versus $\mathbf{z}'$. Several possible conditions have been proposed. Popular choices include (where $y \in \{0, 1\}$ and $\hat{y}$ is the prediction of a machine learning model):

$$P(\hat{y} = y \mid \mathbf{z}) = P(\hat{y} = y \mid \mathbf{z}') \quad \text{(acc)} \quad (1)$$

$$P(\hat{y} = y \mid \mathbf{z}, y = 1) = P(\hat{y} = y \mid \mathbf{z}', y = 1) \quad \text{(TPR)} \quad (2)$$

$$P(\hat{y} = y \mid \mathbf{z}, y = 0) = P(\hat{y} = y \mid \mathbf{z}', y = 0) \quad \text{(TNR)} \quad (3)$$

$$P(\hat{y} = y \mid \mathbf{z}, \hat{y} = 1) = P(\hat{y} = y \mid \mathbf{z}', \hat{y} = 1) \quad \text{(PPV)} \quad (4)$$

$$P(\hat{y} = y \mid \mathbf{z}, \hat{y} = 0) = P(\hat{y} = y \mid \mathbf{z}', \hat{y} = 0) \quad \text{(NPV)} \quad (5)$$

$$P(\hat{y} = 1 \mid \mathbf{z}) = P(\hat{y} = 1 \mid \mathbf{z}') \quad \text{(AR)} \quad (6)$$

Respectively, these consider equality of: (1) accuracy, (2) true positive rate, (3) true negative rate, (4) positive predicted value, (5) negative predicted value, or (6) acceptance rate. Works which use these or related notions include (Hardt et al., 2016; Zafar et al., 2017c;a;b).

In this work we focus on a variant of eq. (6), formulated as a constrained optimization problem by Zafar et al. (2017c) mimicking the $p\%$-rule (Biddle, 2006): for any binary protected attribute $z \in \{0, 1\}$, it aims to achieve

$$\min\left\{\frac{P(\hat{y}=1\,|\,z=1)}{P(\hat{y}=1\,|\,z=0)}, \frac{P(\hat{y}=1\,|\,z=0)}{P(\hat{y}=1\,|\,z=1)}\right\} \geq \frac{p}{100}\ . \quad (7)$$

We believe that in future work, a similar MPC approach could also be used for conditions (1), (2) or (3)—i.e., all the other measures which, to our knowledge, have been addressed with efficient standard (non-private) methods.

2.3. Certification, Training, and Verification

Fairness certification. Given a notion of fairness $\mathbb{F}$, the modeler M would like to work with the regulator REG to obtain a certificate that model $f_{\boldsymbol{\theta}}$ is fair. To do so, we propose that users send their non-sensitive data $\mathbf{D}$ to REG; and send *encrypted* versions of their sensitive data $\mathbf{Z}$ to both M and REG. Neither M nor REG can read the sensitive data. However, we can design a secure protocol between M and REG (described in Section 3) to certify if the model is fair. This setup is shown in Figure 1 (*Left*).

While both REG and M learn the outcome of the certification, we require the following *privacy constraints*: (C1) *privacy of sensitive user data:* no one other than U_i ever learns $\mathbf{z}_i$ in the clear, (C2) *model secrecy:* only M learns $f_{\boldsymbol{\theta}}$ in the clear, and (C3) *minimal disclosure of* $\mathbf{D}$ *to* REG*:* only REG learns $\mathbf{D}$ in the clear.

Fair model training. How can a modeler M learn a fair model without access to users' sensitive data $\mathbf{Z}$? We propose to solve this by having users send their non-sensitive data $\mathbf{D}$ to M and to distribute encryptions of their sensitive data to M and REG as in certification. We shall describe a secure MPC protocol between M and REG to train a fair model $f_{\boldsymbol{\theta}}$ privately. This setup is shown in Figure 1 (*Center*).

Privacy constraints: (C1) privacy of sensitive user data, (C2) model secrecy, and (C3) minimal disclosure of $\mathbf{D}$ to M.

Decision verification. Assume that a malicious M has had model $f_{\boldsymbol{\theta}}$ successfully certified by REG as above. It then swaps $f_{\boldsymbol{\theta}}$ for another potentially unfair model $f_{\boldsymbol{\theta}'}$ in the real world. When a user receives a decision $\hat{y}$, e.g., her mortgage is denied, she can then challenge that decision by asking REG for a verification V. The verification involves M and REG, and consists of verifying that $f_{\boldsymbol{\theta}'}(\mathbf{x}) = f_{\boldsymbol{\theta}}(\mathbf{x})$, where $\mathbf{x}$ is the user's non-sensitive data. This ensures that the user would have been subject to the same result with the certified model $f_{\boldsymbol{\theta}}$, even if $f_{\boldsymbol{\theta}'} \neq f_{\boldsymbol{\theta}}$ and $f_{\boldsymbol{\theta}'}$ is not fair. Hence, while there is no simple technical way to prevent a malicious M from deploying an unfair model, it will get caught if a user challenges a decision that would differ under $f_{\boldsymbol{\theta}}$. This setup is shown in Figure 1 (*Right*).

Privacy constraint: While REG and the user learn the outcome of the verification, we require (C1) privacy of sensitive user data, and (C2) model secrecy.

2.4. Design Choices

We use a regulator for several reasons. Given fair learning is of most benefit to vulnerable individuals, we do not wish to deter adoption with high individual burdens. While MPC could be carried out without the involvement of a regulator, using all users as parties, this comes at a significantly greater computational cost. With current methods, taking that approach would be unrealistic given the size of the user-base in many domains of concern, and would furthermore require all users to be online simultaneously. Introducing a regulator removes these barriers and leaves users' computational burden at a minimum level, with envisaged applications practical with only their web browsers.

In cases where users are uncomfortable sharing $\mathbf{D}$ with either REG or M, it is trivial to extend all three tasks such that all of $\mathbf{x}_i, y_i, \mathbf{z}_i$ remain private throughout, with the computation cost increasing only by a factor of 2. This extension would sometimes be desirable as it restricts the view of M to the final model, prohibiting inferences about $\mathbf{Z}$ when $\mathbf{D}$ is known. However, this setup hinders exploratory data analysis by the modeler which might promote robust model-building, and, in the case of verification, validation by the regulator that user-provided data is correct.

3. Our Solution

Our proposed solution to these three problems is to use Multi-Party Computation (MPC). Before we describe how it can be applied to fair learning, we first present the basic principles of MPC, as well as its limitations particularly in the context of machine learning applications.

3.1. MPC for Machine Learning

Multi-Party Computation protocols allow two parties P_1 and P_2 holding secret values x_1 and x_2 to evaluate an agreed-upon function f, via $y = f(x_1, x_2)$ in a way in which the parties (either both or one of them) learn *only* y. For example, if $f(x_1, x_2) = \mathbb{I}(x_1 < x_2)$, then the parties would learn which of their values is bigger, but nothing else.[1] This corresponds to the well-known *Yao's millionaires problem*: two millionaires want to conclude who is richer without disclosing their wealth to each other. The problem was introduced by Andrew Yao in 1982, and kicked off the area of multi-party computation in cryptography.

[1]The function $\mathbb{I}$ is 1 if its argument is true and 0 otherwise.

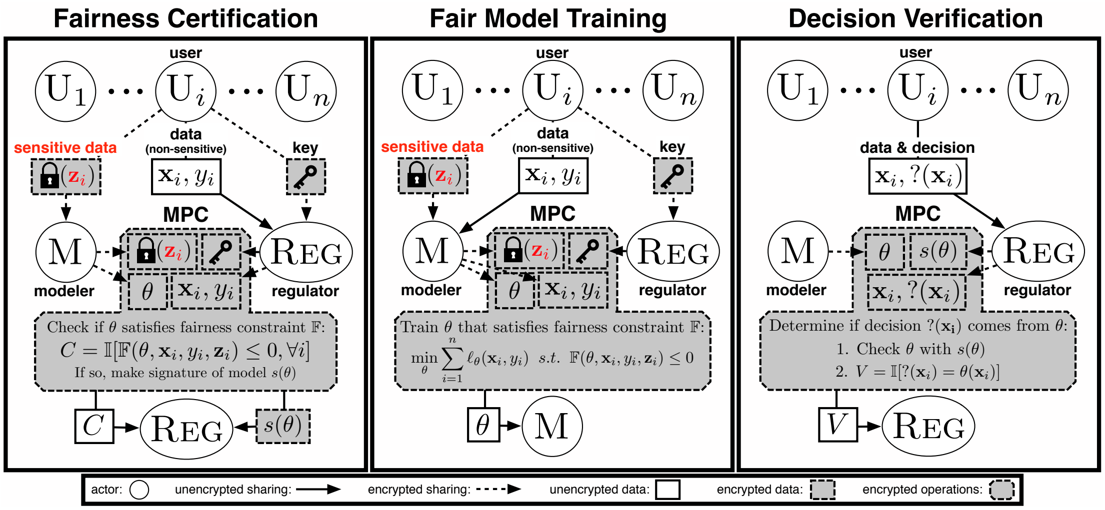

Figure 1. Our setup for *Fairness certification* (*Left*), *Fair model training* (*Center*), and *Decision verification* (*Right*).

In our setting—instead of a simple comparison as in the millionaires problem—f will be either (i) a procedure to check the fairness of a model and certify it, (ii) a machine learning training procedure with fairness constraints, or (iii) a model evaluation to verify a decision. The two parties involved in our computation are the modeler M and the regulator REG. The inputs depend on the case (see Figure 1).

As generic solutions do not yet scale to real-world data analysis tasks, one typically has to tailor custom protocols to the desired functionality. This approach has been followed successfully for a variety of machine learning tasks such as logistic and linear regression (Nikolaenko et al., 2013b; Gascón et al., 2017; Mohassel & Zhang, 2017), neural network training (Mohassel & Zhang, 2017) and evaluation (Juvekar et al., 2018; Liu et al., 2017), matrix factorization (Nikolaenko et al., 2013a), and principal component analysis (Al-Rubaie et al., 2017). In the next section we review challenges beyond scalability issues that arise when implementing machine learning algorithms in MPC.

3.2. Challenges in Multi-Party Machine Learning

MPC protocols can be classified into two groups depending on whether the target function is represented as either a Boolean or arithmetic circuit. All protocols proceed by having the parties jointly evaluate the circuit, processing it gate by gate while keeping intermediate values hidden from both parties by means of a secret sharing scheme. While representing functions as circuits can be done without losing expressiveness, it means certain operations are impractical. In particular, algorithms that execute different branches depending on the input data will explode in size when implemented as circuits, and in some cases lose their run time guarantees (e.g., consider binary search).

Crucially, this applies to *floating-point arithmetic*. While this is work in progress, state-of-the-art MPC floating-point arithmetic implementations take more than 15 milliseconds to multiply two 64 bit numbers (Demmler et al., 2015a, Table 4), which is prohibitive for our applications. Hence, machine learning MPC protocols are limited to *fixed-point* arithmetic. Overcoming this limitation is a key challenge for the field. Another necessity for the feasibility of MPC is to approximate non-linear functions such as the sigmoid, ideally by (piecewise) linear functions.

3.3. Our MPC Protocols

Input sharing. To implement the functionality from Figure 1, we first need a secure procedure for the users to *secret share* a sensitive value, for example her race, with the modeler M and the regulator REG. We use *additive secret sharing*. A value z is represented in a finite domain $\mathbb{Z}_q$—we use $q = 2^{64}$. To share z, the user samples a value r from $\mathbb{Z}_q$ uniformly at random, and sends $z - r$ to M and r to REG. While z can be reconstructed (and subsequently operated on) inside the MPC computation by means of a simple addition, each share on its own does not reveal anything z (other than that it is in $\mathbb{Z}_q$). One can think of arithmetic sharing as a "distributed one-time pad".

In Figure 1, we now reinterpret the key held by REG and the encrypted z by M, as their corresponding shares of the sensitive attributes and denote them by $\langle z \rangle_1$ and $\langle z \rangle_2$ respectively. The idea of privately outsourcing computation to two non-colluding parties in this way is recurrent in MPC, and often referred to as the two-server model (Mohassel & Zhang, 2017; Gascón et al., 2017; Nikolaenko et al., 2013b; Al-Rubaie et al., 2017).

Signing and checking a model. Note that *certification* and *verification* partly correspond to sub-procedures of the *fair training* task: during training we check the fairness constraint $\mathbb{F}$, and repeatedly evaluate partial models on the training dataset (using gradient descent). Hence, *certification* and *verification* do not add technical difficulties over training, which is described in detail in Section 4. However, for verification, we still need to "sign" the model, i.e., REG should obtain a signature $s(\boldsymbol{\theta})$ as a result of model certification, see Figure 1 (*Left*). This signature is used to check in the verification phase, whether a given model $\boldsymbol{\theta}'$ from M satisfies $s(\boldsymbol{\theta}') = s(\boldsymbol{\theta})$ for a certified fair model $\boldsymbol{\theta}$ (in which case $\boldsymbol{\theta} = \boldsymbol{\theta}'$ with high probability). Moreover, we need to preserve the secrecy of the model, i.e., REG should not be able to recover $\boldsymbol{\theta}$ from $s(\boldsymbol{\theta})$. These properties, given that the space of models is large, calls for a cryptographic hash function, such as SHA-256.

Additionally, in our functionality, the hash of $\boldsymbol{\theta}$ should be computed inside MPC, to hide $\boldsymbol{\theta}$ from REG. Fortunately, cryptographic hashes such as SHA-256 are a common benchmark functionality in MPC, and their execution is highly optimized. More concretely, the overhead of computing $s(\boldsymbol{\theta})$, which needs to be done both for certification and verification is of the order of fractions of a second (Keller et al., 2013, Figure 14). While cryptographic hash functions have various applications in MPC, we believe the application to machine learning model certification is novel.

Hence, certification is implemented in MPC as a check that $\boldsymbol{\theta}$ satisfies the criterion $\mathbb{F}$, followed by the computation of $s(\boldsymbol{\theta})$. On the other hand, for verification, the MPC protocol first computes the signature of the model provided by M, and then proceeds with a prediction as long as the computed signature matches the one obtained by REG in the verification phase. An alternative solution is possible based on symmetric encryption under a shared key, as highly efficient MPC implementations of block ciphers such as AES are available (Keller et al., 2017).

Fair training. To realize the *fair training* functionality from the previous section, we follow closely the techniques recently introduced by Mohassel & Zhang (2017). Specifically, we extend their custom MPC protocol for logistic regression to additionally handle linear constraints. This extension may be of independent interest, and has applications for privacy-preserving machine learning beyond fairness. The concrete technical difficulties in achieving this goal, and how to overcome them, are presented in the next section. The formal privacy guarantees of our fair training protocol are stated in the following proposition.

Proposition 1. *For non-colluding* M *and* REG*, our protocol implements the fair model training functionality satisfying constraints (C1)-(C3) in Section 2.3 in the presence of a semi-honest adversary.*

The proof holds in the random oracle model, as a standard simulation argument combining several MPC primitives (Mohassel & Zhang, 2017; Gascón et al., 2017). It leverages security of arithmetic sharing, garbled circuits, and oblivious transfer protocols in the semi-honest model (Goldreich et al., 1987). A general introduction to MPC, as well as a description of the relevant techniques from (Mohassel & Zhang, 2017) used in our protocol, can be found in Section A in the appendix.

4. Technical Challenges of Fair Training

We now present our tailored approaches for learning and evaluating fair models with encrypted sensitive attributes. We do this via the following contributions:

- We argue that current optimization techniques for fair learning algorithms are unstable for fixed-point data, which is required by our MPC techniques.
- We describe optimization schemes that are well-suited for learning over fixed-point number representations.
- We combine tricks to approximate non-linear functions with specialized operations to make fixed-point arithmetic feasible and avoid over- and under-flows.

The optimization problem at hand is to learn a classifier $\boldsymbol{\theta}$ subject to a (often convex) fairness constraint $\mathbb{F}(\boldsymbol{\theta})$:

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \ell_{\boldsymbol{\theta}}(\mathbf{x}_i, y_i) \qquad \text{subject to } \; \mathbb{F}(\boldsymbol{\theta}) \leq \mathbf{0}\,, \tag{8}$$

where $\ell_{\boldsymbol{\theta}}$ is a loss term (the logistic loss in this work). We collect user data from $\mathrm{U}_1, \dots, \mathrm{U}_n$ into matrices $\mathbf{X} \in \mathbb{R}^{n\times d}$, $\mathbf{Z} \in \{0,1\}^{n\times p}$ and a label vector $\mathbf{y} \in \{0,1\}^n$.

Zafar et al. (2017c) use a convex approximation of the p%-rule, see eq. (7), for linear classifiers to derive the constraint:

$$\mathbb{F}(\boldsymbol{\theta}) = \frac{1}{n}|\hat{\mathbf{Z}}^\top \mathbf{X}\boldsymbol{\theta}| - \mathbf{c}\,, \tag{9}$$

where $\hat{\mathbf{Z}}$ is the matrix of all $\hat{\mathbf{z}}_i := \mathbf{z}_i - \bar{\mathbf{z}}$ and $\mathbf{c} \in \mathbb{R}^d$ is a constant vector corresponding to the tightness of the fairness constraint. Here, $\bar{\mathbf{z}}$ is the mean of all inputs $\mathbf{z}_i$. With $\mathbf{A} := {}^1\!/\!_n \hat{\mathbf{Z}}^\top \mathbf{X}$, the p% *constraint* reads $\mathbb{F}(\boldsymbol{\theta}) = |\mathbf{A}\boldsymbol{\theta}| - \mathbf{c}$, where the absolute value is taken element-wise.

4.1. Current Techniques

To solve the optimization problem in eq. (8), with the fairness function $\mathbb{F}$ in eq. (9), Zafar et al. (2017c) use Sequential Least Squares Programming (SLSQP). This technique works by reformulating eq. (8) as a sequence of Quadratic Programs (QPs). After solving each QP, their algorithm uses the Han-Powell method, a quasi-Newton method that iteratively approximates the Hessian $\mathbf{H}$ of the objective function via the update

$$\mathbf{H}_{t+1} = \mathbf{H}_t + \frac{\mathbf{l}_\Delta \mathbf{l}_\Delta^\top}{\boldsymbol{\theta}_\Delta^\top \mathbf{l}_\Delta} - \frac{\mathbf{H}_t \boldsymbol{\theta}_\Delta \boldsymbol{\theta}_\Delta^\top \mathbf{H}_t}{\boldsymbol{\theta}_\Delta^\top \mathbf{H}_t \boldsymbol{\theta}_\Delta}\,,$$

where $\mathbf{l}_\Delta = \mathbf{l}(\boldsymbol{\theta}_{t+1}, \boldsymbol{\lambda}_{t+1}) - \mathbf{l}(\boldsymbol{\theta}_t, \boldsymbol{\lambda}_t)$ and $\mathbf{l}(\boldsymbol{\theta}_t, \boldsymbol{\lambda}_t) = \sum_{i=1}^n \ell_{\boldsymbol{\theta}_t}(\mathbf{x}_i, y_i) + \boldsymbol{\lambda}^\top \mathbb{F}(\boldsymbol{\theta}_t)$ is the Lagrangian of eq. (8). Finally, $\boldsymbol{\theta}_\Delta = \boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t$.

There are two issues with this approach from an MPC perspective. First, solving a sequence of QPs is prohibitively time-consuming in MPC. Second, while the above Han-Powell update performs well on floating-point data, the two divisions by non-constant, non-integer numbers easily underflow or overflow with fixed-point numbers.

4.2. Fixed-Point-Friendly Optimization Techniques

Instead, to solve the optimization problem in eq. (8), we perform stochastic gradient descent and experiment with the following techniques to incorporate the constraints.

Lagrangian multipliers. Here we minimize

$$\mathcal{L} := \frac{1}{n}\sum_{i=1}^{n} \ell_{\boldsymbol{\theta}}^{\mathrm{BCE}}(\mathbf{x}_i, y_i) + \boldsymbol{\lambda}^\top \max\{\mathbb{F}(\boldsymbol{\theta}), \mathbf{0}\}\,,$$

using stochastic gradient descent, i.e., alternating updates $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} \mathcal{L}$ and $\boldsymbol{\lambda} \leftarrow \max\{\boldsymbol{\lambda} + \eta_{\boldsymbol{\lambda}} \nabla_{\boldsymbol{\lambda}} \mathcal{L}, \mathbf{0}\}$, where $\eta_{\boldsymbol{\theta}}, \eta_{\boldsymbol{\lambda}}$ are the learning rates.

Projected gradient descent. For this method, consider specifically the $p\%$-rule based notion $\mathbb{F}(\boldsymbol{\theta}) = |\mathbf{A}\boldsymbol{\theta}| - \mathbf{c}$. We first define $\hat{\mathbf{A}}$ as the matrix consisting of the rows of $\mathbf{A}$ for which $\mathbb{F}(\boldsymbol{\theta}) > \mathbf{0}$, i.e., where the constraint is active. In each step, we project the computed gradient of the binary-cross-entropy loss $\mathcal{L}^{\mathrm{BCE}}$—either of a single example or averaged over a minibatch—back into the constraint set, i.e.,

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_{\boldsymbol{\theta}} (\mathrm{Id}_d - \hat{\mathbf{A}}^\top (\hat{\mathbf{A}}\hat{\mathbf{A}}^\top)^{-1} \hat{\mathbf{A}}) \nabla_{\boldsymbol{\theta}} \ell_{\boldsymbol{\theta}}^{\mathrm{BCE}}\,. \quad (10)$$

Interior point log barrier (Boyd & Vandenberghe, 2004). We can approximate eq. (8) for the $p\%$-rule constraint $\mathbb{F}(\boldsymbol{\theta}) = |\mathbf{A}\boldsymbol{\theta}| - \mathbf{c}$ by: minimize $\sum_{i=1}^n \ell_{\boldsymbol{\theta}}^{\mathrm{BCE}}(\mathbf{x}_i, y_i) - \frac{1}{t}\sum_{j=1}^p \big(\log(\mathbf{a}_j^\top \boldsymbol{\theta} + c_j) + \log(-\mathbf{a}_j^\top \boldsymbol{\theta} + c_j)\big)$, where $\mathbf{a}_j$ is the jth row of $\mathbf{A}$. The parameter t trades off the approximation of the true objective ($I_-(u) = 0$ for $u \leq 0$ and $I_-(u) = \infty$ for $u > 0$) and the smoothness of the objective function. Throughout training t is increased, allowing the solution to move closer to the boundary. As the gradient of the objective has a simple closed form representation, we can perform regular (stochastic) gradient descent.

After extensive experiments (see Section 5) we found the Lagrangian multipliers technique to work best, both in yielding high accuracies, reliably staying within the constraints and being robust to hyperparameter changes such as learning rates or the batch size. For a proof of concept, in Section 5 we focus on the $p\%$-rule, i.e., eq. (9). Note that the gradients for eq. (2) and eq. (3) take a similarly simple form, i.e., balancing the true positive or true negative rates (corresponding to equal opportunity or equal odds) is simple to implement for the Lagrangian multiplier technique, but harder for projected gradient descent. However, these fairness notions are more expensive as we have to compute $\mathbf{Z}^\top\mathbf{X}$ for each update step, instead of pre-computing it once at the beginning of training, see Algorithm 1 in the appendix. We could speed up the computation again by evaluating the constraint only on the current minibatch for each update, in which case we risk violating the fairness constraint.

MPC-friendliness. For eq. (9), we can compute the gradient updates in all three methods with elementary linear algebra operations (matrix multiplications) and a single evaluation of the logistic function. While MPC is well suited for linear operations, most nonlinear functions are prohibitively expensive to evaluate in an MPC framework. Hence we tried two piecewise linear approximations for $\sigma(x)$. The first was recently suggested for machine learning in an MPC context (Mohassel & Zhang, 2017) and is simply constant 0 and 1 for $x < -0.5$ and $x > 0.5$ respectively, and linear in between. The second uses the optimal first order Chebychev polynomial on each interval $[x, x+1]$ for $x \in \{-5, -4, \ldots, 4\}$, and is constant 0 or 1 outside of $[-5, 5]$ (Faiedh et al., 2001). While it is more accurate, we only report results for the simpler first approximation, as it yielded equal or better results in all our experiments.

As the largest number that can be represented in fixed-point format with m integer and m fractional bits is roughly $2^m + 1$, overflow becomes a common problem. Since we whiten the features $\mathbf{X}$ column-wise, we need to be careful whenever we add roughly 2^m numbers or more, because we cannot even represent numbers greater than 2^m. In particular, the minibatch size has to be smaller than this limit. For large n, the multiplication $\mathbf{Z}^\top\mathbf{X}$ in the fairness function $\mathbb{F}$ for the $p\%$-rule is particularly problematic.

Hence, we split both factors into blocks of size $b \times b$ with $b < 2^m$ and normalize the result of each blocked matrix multiplication by b before adding up the blocks. We then multiply the sum by $b/n > 2^{-m}$. As long as b, b/n (and thus also n/b) can be represented with sufficient precision, which is the case in all our experiments, this procedure avoids under- and overflow. Note that we require the sample size n to be a multiple of b. In practice, we have to either discard or duplicate part of the data. Since the latter may introduce bias, we recommend subsampling. Once we have (an approximation of) $\mathbf{A} \in \mathbb{R}^{p \times d}$, we resort to normal matrix multiplication, as typically $p, d \lesssim 100$, see Table 1.

Division is prohibitively expensive in MPC. Hence, we set the minibatch size to a power of two, which allows us to use fast bit shifts for divisions when averaging over minibatches. To exploit the same trick when averaging over/across blocks in the blocked matrix multiplication, we choose n as the largest possible power of two, see Table 1.

Table 1. Dataset sizes and online timing results of MPC certification and training over 10 epochs with batch size 64.

	Adult	Bank	COMPAS	German	SQF
n training examples	2^{14}	2^{15}	2^{12}	2^{9}	2^{16}
d features	51	62	7	24	23
p sensitive attr.	1	1	7	1	1
certification	802 ms	827 ms	288 ms	250 ms	765 ms
training	43 min	51 min	7 min	1 min	111 min

Algorithm 1 in Section B in the appendix describes the computations M and REG have to run for fair model training using the Lagrangian multiplier technique and the $p\%$-rule from eq. (9). We implicitly assume all computations are performed jointly on additively shared secrets.

5. Experiments

The root cause for most technical difficulties pointed out in the previous section is the necessity to work with fixed-point numbers and the high computational cost of MPC. Hence, major concerns are loss of precision and infeasible running times. In this section, we show how to overcome both doubts and that fair training, certification and verification are feasible for realistic datasets.

5.1. Experimental Setup and Datasets

We work with two separate code bases. Our Python code does not implement MPC, to be able to flexibly switch between floating and fixed-point numbers as well as exact non-linear functions and their approximations. We use it mostly for validation and empirical guidance in our design choices. The full MPC protocol is implemented in C++ on top of the Obliv-C garbled circuits framework (Zahur & Evans, 2015a) and the Absentminded Crypto Kit (lib). This is done as described in Section 3 for the Lagrangian multiplier technique (see Section A in the appendix for more details). It accurately mirrors the computations performed by the first implementation on encrypted data.[2] Except for the timing results in Table 1, all comparisons with floating-point numbers or non-linearities were done with the versatile Python implementation. Details about parameters and the algorithm can be found in Section B in the appendix.

We consider 5 real world datasets, namely the adult (*Adult*), German credit (*German*), and bank market (*Bank*) datasets from the UCI machine learning repository (Lichman, 2013), the stop, question and frisk 2012 dataset (*SQF*),[3] and the COMPAS dataset (Angwin et al., 2016) (*COMPAS*). For practical purposes (see Section 4), we subsample 2^i examples from each dataset with the largest possible i, see Table 1. Moreover, we also run on synthetic data, generated as described by Zafar et al. (2017c, Section 4.1), as it allows us to control the correlation between the sensitive attributes and the class labels. It is thus well suited to observe how different optimization techniques handle the fairness-accuracy trade off. For comparison we use the SLSQP approach described in Section 4.1 as a baseline. We run all methods for a range of constraint values in $[10^{-4}, 10^{0}]$ and a corresponding range for SLSQP.

In the plots in this section, discontinuations of lines indicate failed experiments. The most common reasons are overflow and underflow for fixed-point numbers, and instability due to exploding gradients. Plots and analyses for the remaining datasets can be found in Section C in the appendix.

5.2. Comparing Optimization Techniques

First we evaluate which of the three optimization techniques works best in practice. Figure 2 shows the test set accuracy over the constraint value. By design, the synthetic dataset exhibits a clear trade-off between accuracy and fairness. The Lagrange technique closely follows the (dotted) baseline from (Zafar et al., 2017c), whereas iplb performs slightly worse (and fails for small c). Even though the projected gradient method formally satisfies the proxy constraint for the $p\%$ rule, it does so by merely shrinking the parameter vector $\boldsymbol{\theta}$, which is why it also fails for small c. We analyze this behavior in more detail in Section C in the appendix.

The COMPAS dataset is the most challenging as it contains 7 sensitive attributes, one of which has only 10 positive instances in the training set. Since we enforce the fairness constraint individually for each sensitive attribute (we randomly picked one for visualization), the classifier tends to collapse to negative predictions. All three methods maintain close to optimal accuracy in the unconstrained region, but collapse more quickly than SLSQP. This example shows that the $p\%$-rule proxy itself needs careful interpretation when applied to multiple sensitive attributes simultaneously and that our SGD based approach seems particularly prone to collapse in such a scenario. On the Bank dataset accuracy increases for iplb and Lagrange when the constraint becomes active as c decreases until they match the baseline. Determining the cause of this—perhaps unintuitive—behavior requires further investigation. We currently suspect the constraint to act as a regularizer. The projected gradient method is unreliable on the Bank dataset.

Empirically, the Lagrangian multiplier technique is most robust with maximal deviations of accuracy from SLSQP of $< 4\%$ across the 6 datasets and all constraint values. We substantiate this claim in Section C of the appendix. For the rest of this section we only report results for Lagrangian multipliers. Figure 2 also shows that using a piecewise linear approximation as described in Section 4 for the logistic function does not spoil performance.

[2] Code is available at https://github.com/nikikilbertus/blind-justice

[3] https://perma.cc/6CSM-N7AQ

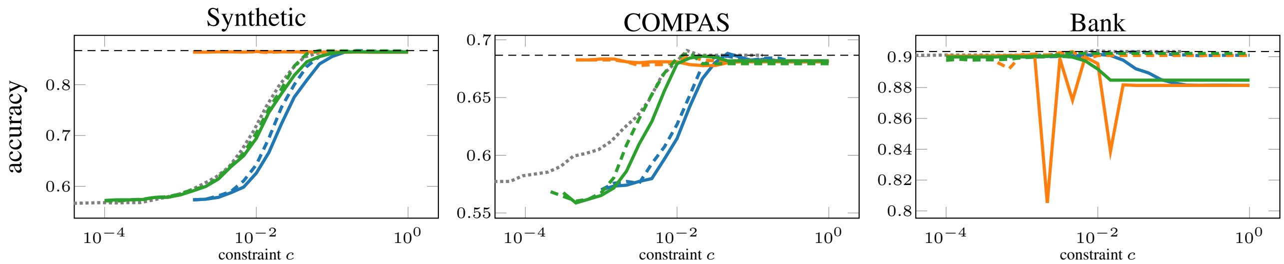

Figure 2. Test set accuracy over the $p\%$ value for different optimization methods (blue: iplb, orange: projected, green: Lagrange) and either no approximation (*continuous*) or a piecewise linear approximation (*dashed*) of the sigmoid using floating-point numbers. The gray dotted line is the baseline (see Section 4.1) and the black dashed line is unconstrained logistic regression (from scikit-learn).

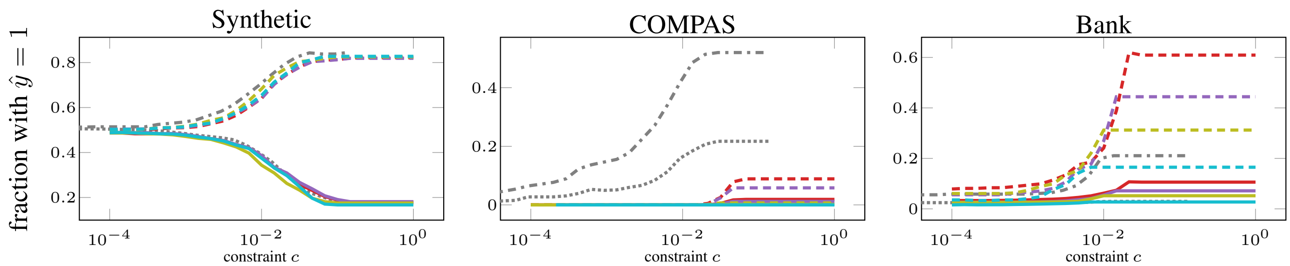

Figure 3. The fraction of people with $z = 0$ (*continuous/dotted*) and $z = 1$ (*dashed/dash-dotted*) who get assigned positive outcomes (red: no approx. + float, purple: no approx. + fixed, yellow: pw linear + float, turquoise: pw linear + fixed, gray: baseline).

5.3. Fair Training, Certification and Verification

Figure 3 shows how the fractions of users with positive outcomes in the two groups ($z = 0$ is continuous and $z = 1$ is dashed) are gradually balanced as we decrease the fairness constraint c. These plots can be interpreted as the degree to which disparate impact is mitigated as the constraint is tightened. The effect is most pronounced for the synthetic dataset by construction. As discussed above, the collapse for the COMPAS dataset occurs faster than for SLSQP due to the constraints from multiple sensitive attributes. In the Bank dataset, for large c—before the constraint becomes active—the fractions of positive outcomes for $z = 1$ differ, which is related to the slightly suboptimal accuracy at large c that needs further investigation. However, as the constraint becomes active, the fractions are balanced at a similar rate as the baseline. Overall, our Lagrangian multiplier technique with fixed point numbers and piecewise linear approximations of non-linearities robustly manages to satisfy the p%-rule proxy at similar rates as the baseline with only minor losses in accuracy on all but the challenging COMPAS dataset.

In Table 1 we show the online running times of 10 training epochs on a laptop computer. While training takes several orders of magnitudes longer than a non-MPC implementation, our approach still remains feasible and realistic. We use the one time offline precomputation of multiplication triples described and timed in Mohassel & Zhang (2017, Table 2). As pointed out in Section 3, certification of a trained model requires checking whether $\mathbb{F}(\boldsymbol{\theta}) > 0$. We already perform this check at least once for each gradient update during training. It only takes a negligible fraction of the computation time, see Table 1. Similarly, the operations required for certification stay well below one second.

Discussion. In this section, we have demonstrated the practicability of private and fair model training, certification and verification using MPC as described in Figure 1. Using the methods and tricks introduced in Section 4, we can overcome accuracy as well as over- and underflow concerns due to fixed-point numbers. Offline precomputation combined with a fast C++ implementation yield viable running times for reasonably large datasets on a laptop computer.

6. Conclusion

Real world fair learning has suffered from a dilemma: in order to enforce fairness, sensitive attributes must be examined; yet in many situations, users may feel uncomfortable in revealing these attributes, or modelers may be legally restricted in collecting and utilizing them. By introducing recent methods from MPC, and extending them to handle linear constraints as required for various notions of fairness, we have demonstrated that it is practical on real-world datasets to: (i) certify and sign a model as fair; (ii) learn a fair model; and (iii) verify that a fair-certified model has indeed been used; all while maintaining cryptographic privacy of all users' sensitive attributes. Connecting concerns in privacy, algorithmic fairness and accountability, our proposal empowers regulators to provide better oversight, modelers to develop fair and private models, and users to retain control over data they consider highly sensitive.

Acknowledgments

The authors would like to thank Chris Russell and Phillipp Schoppmann for useful discussions and help with the implementation, as well as the anonymous reviewers for helpful comments. AG and MK were supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1. MV was supported by EPSRC grant EP/M507970/1. AW acknowledges support from the David MacKay Newton research fellowship at Darwin College, The Alan Turing Institute under EPSRC grant EP/N510129/1 & TU/B/000074, and the Leverhulme Trust via the CFI.

References

Absentminded crypto kit. https://bitbucket.org/jackdoerner/absentminded-crypto-kit.

Al-Rubaie, M., Wu, P. Y., Chang, J. M., and Kung, S. Privacy-preserving PCA on horizontally-partitioned data. In *DSC*, pp. 280–287. IEEE, 2017.

Angwin, J., Larson, J., Mattu, S., and Kirchner, L. Machine bias: There is software used across the country to predict future criminals. and it is biased against blacks. *ProPublica, May*, 23, 2016.

Barocas, S. and Selbst, A. D. Big data's disparate impact. *California Law Review*, 104:671–732, 2016.

Bennett Capers, I. Blind justice. *Yale Journal of Law & Humanities*, 24:179, 2012.

Biddle, D. *Adverse impact and test validation: A practitioner's guide to valid and defensible employment testing*. Gower Publishing, Ltd., 2006.

Boyd, S. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Calders, T. and Žliobaitė, I. Why unbiased computational processes can lead to discriminative decision procedures. In *Discrimination and Privacy in the Information Society*, pp. 43–59. Springer, 2012.

Damgård, I., Pastro, V., Smart, N. P., and Zakarias, S. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pp. 643–662. Springer, 2012.

Demmler, D., Dessouky, G., Koushanfar, F., Sadeghi, A., Schneider, T., and Zeitouni, S. Automated synthesis of optimized circuits for secure computation. In *ACM Conference on Computer and Communications Security*, pp. 1504–1517. ACM, 2015a.

Demmler, D., Schneider, T., and Zohner, M. ABY – a framework for efficient mixed-protocol secure two-party computation. In *NDSS*. The Internet Society, 2015b.

Dwork, C., Hardt, M., Pitassi, T., Reingold, O., and Zemel, R. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pp. 214–226. ACM, 2012.

Faiedh, H., Gafsi, Z., and Besbes, K. Digital hardware implementation of sigmoid function and its derivative for artificial neural networks. *Proceeding of the 13th International Conference on Microelectronics, 2001.*, pp. 189 – 192, 11 2001.

Feldman, M., Friedler, S., Moeller, J., Scheidegger, C., and Venkatasubramanian, S. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 259–268, 2015.

Fredrikson, M., Jha, S., and Ristenpart, T. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1322–1333, 2015.

Friedler, S. A., Scheidegger, C., and Venkatasubramanian, S. On the (im)possibility of fairness. 2016. arXiv:1609.07236v1 [cs.CY].

Gascón, A., Schoppmann, P., Balle, B., Raykova, M., Doerner, J., Zahur, S., and Evans, D. Privacy-Preserving Distributed Linear Regression on High-Dimensional Data. *Proceedings on Privacy Enhancing Technologies*, 2017 (4):345–364, October 2017.

Goldreich, O. *The Foundations of Cryptography – Volume 2, Basic Applications*. Cambridge University Press, 2004.

Goldreich, O., Micali, S., and Wigderson, A. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pp. 218–229. ACM, 1987.

Graham, C. NHS cyber attack: Everything you need to know about 'biggest ransomware' offensive in history. *Telegraph, May 20*, 2017.

Grgić-Hlača, N., Zafar, M. B., Gummadi, K. P., and Weller, A. Beyond distributive fairness in algorithmic decision making: Feature selection for procedurally fair learning. In *AAAI*, 2018.

Handel, P., Skog, I., Wahlstrom, J., Bonawiede, F., Welch, R., Ohlsson, J., and Ohlsson, M. Insurance telematics: Opportunities and challenges with the smartphone solution. *IEEE Intelligent Transportation Systems Magazine*, 6(4):57–70, 2014.

Hardt, M., Price, E., and Srebro, N. Equality of opportunity in supervised learning. In *NIPS*, 2016.

Juvekar, C., Vaikuntanathan, V., and Chandrakasan, A. Gazelle: A Low Latency Framework for Secure Neural Network Inference. *IACR Cryptology ePrint Archive*, 2018:73, 2018.

Keller, M., Scholl, P., and Smart, N. P. An architecture for practical actively secure MPC with dishonest majority. In *ACM Conference on Computer and Communications Security*, pp. 549–560. ACM, 2013.

Keller, M., Orsini, E., Rotaru, D., Scholl, P., Soria-Vazquez, E., and Vivek, S. Faster secure multi-party computation of AES and DES using lookup tables. In *ACNS*, volume 10355 of *Lecture Notes in Computer Science*, pp. 229–249. Springer, 2017.

Keller, M., Pastro, V., and Rotaru, D. Overdrive: Making SPDZ great again. In *EUROCRYPT (3)*, volume 10822 of *Lecture Notes in Computer Science*, pp. 158–189. Springer, 2018.

Kroll, J. A., Huey, J., Barocas, S., Felten, E. W., Reidenberg, J. R., Robinson, D. G., and Yu, H. Accountable algorithms. *University of Pennsylvania Law Review*, 165, 2016.

Lichman, M. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Lindell, Y. How To Simulate It – A Tutorial on the Simulation Proof Technique. *IACR Cryptology ePrint Archive*, 2016:46, 2016.

Liu, J., Juuti, M., Lu, Y., and Asokan, N. Oblivious neural network predictions via minionn transformations. In *CCS*, pp. 619–631. ACM, 2017.

Mohassel, P. and Zhang, Y. SecureML: A system for scalable privacy-preserving machine learning. In *IEEE Symposium on Security and Privacy (SP)*, pp. 19–38, 2017.

Nikolaenko, V., Ioannidis, S., Weinsberg, U., Joye, M., Taft, N., and Boneh, D. Privacy-preserving matrix factorization. In *ACM Conference on Computer and Communications Security*, pp. 801–812. ACM, 2013a.

Nikolaenko, V., Weinsberg, U., Ioannidis, S., Joye, M., Boneh, D., and Taft, N. Privacy-preserving ridge regression on hundreds of millions of records. In *IEEE Symposium on Security and Privacy*, pp. 334–348. IEEE Computer Society, 2013b.

Schreurs, W., Hildebrandt, M., Kindt, E., and Vanfleteren, M. *Cogitas, Ergo Sum*. The Role of Data Protection Law and Non-discrimination Law in Group Profiling in the Private Sector. In *Profiling the European Citizen*, pp. 241–270. Springer, 2008.

Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., and Ristenpart, T. Stealing machine learning models via prediction apis. In *USENIX Security Symposium*, pp. 601–618, 2016.

Veale, M. and Binns, R. Fairer machine learning in the real world: Mitigating discrimination without collecting sensitive data. *Big Data & Society*, 4(2), 2017.

Veale, M. and Edwards, L. Clarity, Surprises, and Further Questions in the Article 29 Working Party Draft Guidance on Automated Decision-Making and Profiling. *Computer Law & Security Review*, 2018. doi: 10.1016/j.clsr.2017.12.002.

Yao, A. C.-C. How to Generate and Exchange Secrets (Extended Abstract). In *FOCS*, pp. 162–167. IEEE Computer Society, 1986.

Zafar, M. B., Valera, I., Gomez-Rodriguez, M., and Gummadi, K. P. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *WWW*, 2017a.

Zafar, M. B., Valera, I., Rodriguez, M., Gummadi, K., and Weller, A. From parity to preference-based notions of fairness in classification. In *Advances in Neural Information Processing Systems*, pp. 228–238, 2017b.

Zafar, M. B., Valera, I., Rodriguez, M. G., and Gummadi, K. P. Fairness Constraints: Mechanisms for Fair Classification. In *AISTATS*, 2017c.

Zahur, S. and Evans, D. Obliv-c: A language for extensible data-oblivious computation. *IACR Cryptology ePrint Archive*, 2015:1153, 2015a.

Zahur, S. and Evans, D. Obliv-C: A Language for Extensible Data-Oblivious Computation. *IACR Cryptology ePrint Archive*, 2015:1153, 2015b.

Žliobaitė, I. and Custers, B. Using sensitive personal data may be necessary for avoiding discrimination in data-driven decision models. *Artificial Intelligence and Law*, 24(2):183–201, 2016.